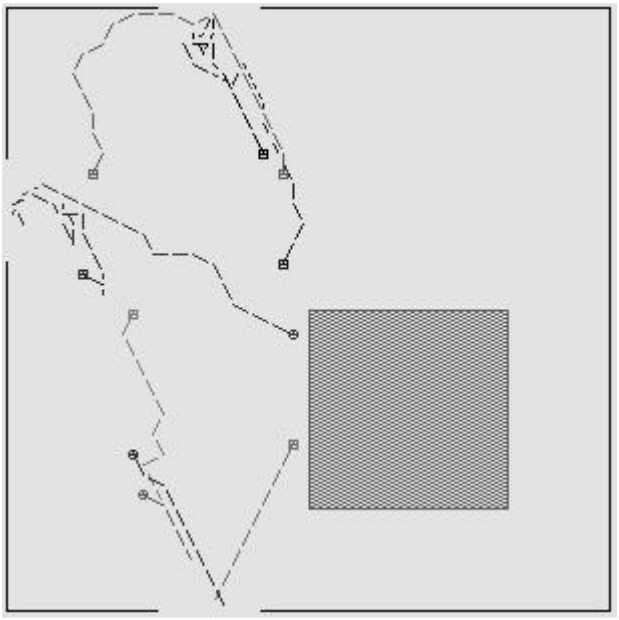
GTICC

P-Space

Version 3.3



User manual

February 2000

Grup de Tecnologia Informàtica en Ciències del Comportament
Computer Technology in the Behavioral Science Research Group
University of Barcelona, Spain

Contents

Introduction 2

Installation 4

Running 5

Hot keys 7

Commands 8

Examples 29

Output 31

Errors 33

Introduction

P-Space is a program that stimulates spatial behavior in a group of *agents* (humans, robots, etc). Agents have attitudes towards each other, which determine the constant or *variable ideal distances* they want to keep from other agents. Agents live in a universe called *room*, which they can exit and re-enter (if *doors* exist). The room can contain *objects,* or areas unavailable to agents. The room is divided into *cells,* and time is measured discretely. A cell cannot be occupied simultaneously by more than one agent. At each time unit, each agent is at the center of its own *neighborhood* of cells, an area surrounding the agent and limiting the cells where it will be able to move to at the next time unit. When agents move, their neighborhoods move with them.

An agent moves to that cell in its current neighborhood that will minimize its *social dissatisfaction*. An agent's dissatisfaction is defined as a weighted function of the discrepancy between possible future distances, as estimated by the agent, and ideal distances from each other agent. The ideal dissatisfaction (the one the agent wishes to get) is its minimum possible dissatisfaction before moving, and the real dissatisfaction is the one it has after moving. These dissatisfactions can differ because the other agents might have moved to other cells in their respective neighborhoods, and thus the agent will not reach its ideal dissatisfaction. The difference between real and ideal dissatisfaction is the agent's *frustration*.

At each time unit agents decide to move within their respective neighborhoods *simultaneously* and *independently*. However, if two or more neighborhoods overlap, then some of their cells may be candidates for more than one agent if they happen to provide minimum dissatisfactions for all of them. At each time unit, agents' priorities to move are sorted randomly. Then, agents with lower priorities will be able to move only to those cells not chosen by agents with higher priorities.

Agents have individual features which determine their *attention scopes,* their *tendencies to exit* the room, their *tendencies to remain outside*, and the effect that real distances from other agents have on the discrepancies between real and ideal distances (agent *gravity*).

Ideal distances can be neutral, constant, increasing or decreasing as a function of time, random, a function of time in proximity between the agents, or, generally, a function of their interactions (*ideal distance models*). Ideal distances from agents to objects and from agents to doors can be defined as well. Different models can be defined for each agent-to-agent, agent-to-object, and agent-to-door interaction.

As agents move to those cells in their neighborhoods where dissatisfactions tend to minimize (that is to say, where real and ideal distances are most similar), they will keep moving around the room until their dissatisfactions are null. Nevertheless, it may be that, due to the ideal

distances of other agents, the null level of dissatisfaction is unreachable. In this case the group as a system can reach either static or dynamical stability (agents stop moving or keep moving indefinitely). Likewise, if agents get in and out of the room, or if ideal distance models that are functions of time in proximity have been defined, then steady states or a dynamic equilibria can be reached, and be broken periodically or chaotically.

Installation

After downloading P-Space, copy file PSPACExx.ZIP (where xx indicates the program version) to a directory in your system and unpack it using Pkunzip or WinZip. Program P-Space consists of the following files:

PSPACE.EXE	Executable
PSPENGxx.TRB	English message file
PSPESPxx.TRB	Spanish message file
PSPCATxx.TRB	Catalan message file
PSPENGxx.HLP	English help file
PSPESPxx.HLP	Spanish help file
PSPCATxx.HLP	Catalan help file
PSPESPxx.NAV	English help navigation file
PSPESPxx.NAV	Spanish help navigation file
PSPCATxx.NAV	Catalan help navigation file
V.EXE	File browser
VENG.TRB	File browser English message file
VESP.TRB	File browser Spanish message file
VCAT.TRB	File browser Catalan message file
EGAVGA.BGI	Borland DOS graphical interface file
LITT.CHR	Borland graphical font file

All the files must be in the same directory.

Running

P-Space 3.3 runs in DOS or in a Windows 98 window. In order to run P-Space, you must first write down commands specifying the number of agents, their features, the room size, the ideal distance models, and so forth, and save them as an ASCII file. Commands must follow a specific P-Space syntax or set of conventions (described below):

To run the program, change to the P-Space directory and type one of the following commands:

`PSPACE`	The program prompts for a command file name, and simulates according to the commands specified in the file. Default system language is used (English, Spanish, or Catalan).
`PSPACE –i`	Ditto in English
`PSPACE -e`	Ditto in Spanish
`PSPACE –c`	Ditto in Catalan
`PSPACE <file>`	Simulates according to the commands specified in the file. Default system language is used (English, Spanish, or Catalan).
`PSPACE <file> -i`	Simulates according to the commands specified in the file. Commands must be written in English.
`PSPACE <file> -e`	Simulates according to the commands specified in the file. Commands must be written in Spanish.
`PSPACE <file> -c`	Simulates according to the commands specified in the file. Commands must be written in Catalan.
`PSPACE -?`	Displays help text, using default system language. If file browser V.EXE is in the same directory, then P-Space will call it in order to display the help, otherwise help will be simple printed to the screen. After displaying help, the program will prompt for a command file (if you do not want to run any simulation, press Ctrl-C at the prompt).
`PSPACE -? -i`	Ditto in English
`PSPACE -? -e`	Ditto in Spanish
`PSPACE -? –c`	Ditto in Catalan

Default language used by the program is the OS language. However, if P-Space fails to detect the OS language, then either it must be forced to run using the desired language (with running options `–i, -e, -c`), or commands must be written in the language that P-Space recognizes as default.

The help displayed by the program on screen are the contents of the HLP files. They are ASCII files, and can be read or printed by any text editor or word processor. Their contents are a summary of this user manual. When in the file browser V.EXE, press F3 to navigate the help text.

Hot keys

While the program is running in graphical mode (see *OUTPUT*), the folowing hot keys can be used:

- **R** (Run): Starts the simulation, or resumes it if it was stopped.
- **S** (Step): Stops the simulation, and executes one step or time unit. Pressing **R** again resumes the Run mode.
- **+**: Increases execution speed (i.e., reduces the pause between consecutive steps; see *PAUSE*).
- **-**: Decreases execution speed (i.e., increases the pause between consecutive steps; see *PAUSE*).
- **C** (Chaos): Inserts chaos in the movements of the agents, provoking that they move randomly within their neighborhoods at the current time unit (see *CHAOS*).
- **E** (End): Ends the simulation. Pressing **E** twice terminates the program. If more than one simulation was requested in the command file (see *NSIMULATIONS*), then pressing **R** after **E** starts the next simulation.

Commands

In order to carry out a simulation, P-Space must read commands from an ASCII command file, which can be created using any text editor.

- Only the *IDEAL* command is mandatory, the rest are optional. If they are omitted, default parameters for them are used. Except for specific cases, commands may be entered in any sequence.
- Commands must fit in a single line of text. Only the *IDEAL* command may expand more than one line.
- Only the first four letters of command names and alphanumerical parameters are necessary.
- Command and parameters names are case insensitive.
- Comments can be inserted in the command file by adding a % sign in the first column, which indicates that the rest of the line must be ignored. Therefore, in order to cancel out a command without deleting it, just precede the command with a % sign.
- In the specifications that follow, { } enclose items that are optional and can be specified progressively, that is { *a b c d* } means that *a, b, c,* and *d* may be omitted; or that *a* may be specified, and *b, c,* and *d* omitted; or that *a* and *b* can be specified, and *c* and *d* omitted; or that *a, b,* and *c* may be specified, and *d* omitted; or that *a, b, c,* and *d* may be specified. Square brackets [ ] enclose items that are mandatory. A vertical bar separates mutually exclusive items.

NSIMULATIONS *n*

Specifies that the process must be repeated *n* times. Note that if you request that results be saved to a file, a large *n* usually generates a huge file (see *OUTPUT*). Default is *n* = 1.

TIME *t*

Specifies the total number of steps or discrete time units for the simulation. Total number of steps will equal *t* by the number of simulations. Note that if you request that results be saved to a file, its number of lines will equal the total number of steps at least (see *OUTPUT*). Default is *t* = 100.

ROOM *x y*

Defines the size of the room where agents will move. The room is a rectangle with horizontal side *x* and vertical side *y,* where *x* and *y* are expressed in number of cells. A cell is represented on the screen by a square of 5 by 5 pixels. “Horizontal” (or width) and “vertical” (or depth) refer to the room location on screen. Walls are called *north, east, south* and *west*. Maximum room size is x = *120, y = 90*, and minimum is *x = y = 10.* If x ≤ 80, and graphical output is requested

(see *OUTPUT*), then two additional graphical representations are shown: density of occupation, and a plot of variables chosen by user (see *SQUARE*). If $y \leq 70$, then an additional plot specified with the *RECTANGLE* command is displayed below the room. Default is $x = 30$, $y = 30$. Figure 1 shows a P-Space sample screen.

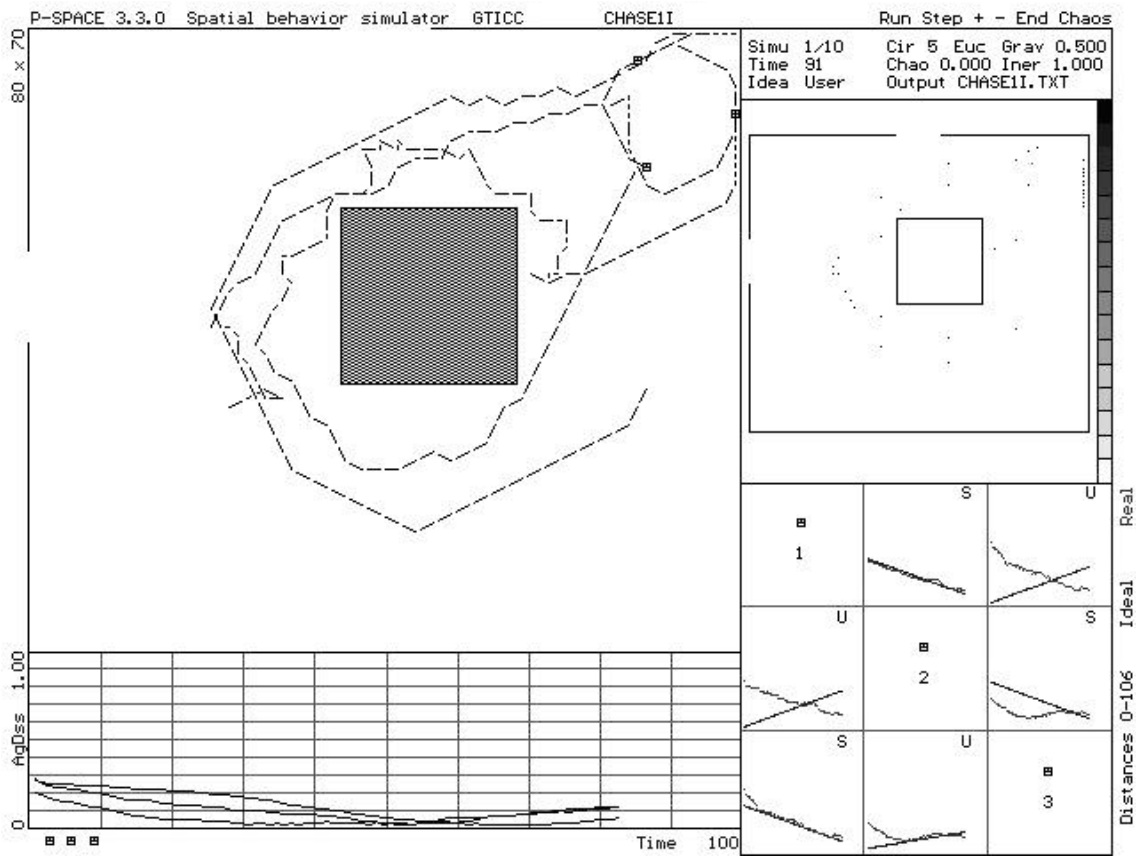


Figure 1. Sample screen of a P-Space simulation. A room with three interacting agents, one object, and two doors is shown in the upper left area. The upper right section of the screen displays the room to scale, where frequency of occupation is represented using color codes in a real screen. The lower left plot shows the agents' real dissatisfactions as a function of time. Ideal and real distances between agents as a function of time are shown in the lower right 3x3 grid. Actual graphical representations may vary depending on the room size specified, and on the variables selected for the lower right (see *SQUARE*) and lower left (see *RECTANGLE*) areas.

NAGENTS *n*

Defines the number of agents. Maximum is 20 and minimum is 1. The greater the number of agents, the slower the simulation. Default is *n = 10*. If agent features are specified by means of *AGENT* or *BLOCK* commands, then *NAGENTS* may be omitted. If *NAGENTS* is specified, then it must precede any *AGENT* or *BLOCK* command. *NAGENTS* is useful for specifying a number of agents different from the default one, while accepting default parameters for the agents' features. If NAGENTS is specified but not *AGENT* or *BLOCK*, then initial agent coordinates will be randomly selected. When an output file is requested (see *OUTPUT*), its size increases exponentially as the number of agent increases.

AGENT *x y* { *heading angle limit timeout*
IN | OUT SAME | RANDOM EXIT | NOEXIT }

Defines an agent's individual features:

- *x, y* are its initial coordinates within the room, measured in number of cells to the right and downwards from the origin of coordinates (0,0), the cell in the upper left corner of the room (see Figure 2). If x = -1*, y* = -1, or both, then initial coordinates will be randomly selected. If x = -2, *y* = -2*,* then the agent will be initially out of the room, which is equivalent to specifying the *OUT* parameter.
- *heading* is the agent's initial heading, measured in degrees from 0 to 360 counterclockwise from axis *X*. For an agent facing the east, north, west, and south walls, heading equals 0, 90, 180 and 270 degrees respectively. Once the simulation has started, a vector joining its positions at times *t*-1 and *t* defines the agent's heading. By default, initial heading is selected randomly.
- *angle* is the agent's attention scope angle, measured in degrees and centered at the vector defining its current heading. Default is 360 degrees.
- *limit* is the maximum distance to which the agent pays attention. Default is 1000. *Angle* and *limit* define a circular sector that is the agent's attention scope. The agent ignores the area within the room that is outside its current attention scope. Therefore, the agent acts as if its ideal distance to agents, objects, and doors lying in that area is neutral (in Figure 2, the white area represents the agent's current attention scope).
- *timeout* is the mathematical expectancy of the amount of time that the agent will remain out of the room after exiting it, or before entering it for the first time if it was initially out. Actual time out is computed by sampling from an exponential distribution whose mathematical expectancy equals *timeout.* Default is 0.
- *IN* or *OUT* specify the agent's place when the simulation starts. If *OUT* is specified, then the initial coordinates are dummy. If *OUT* is specified and the room has no doors (see *DOOR*), then the agent will remain out of the room throughout the simulation. Default is *IN.*

- *SAME* or *RANDOM* define the door used by the agent for reentering the room: either the same door through which it exited the room previously or a door selected at random. If there are no doors, then this parameter is dummy. Default is *SAME*.
- *EXIT* or *NOEXIT* define the agent's readiness to exit the room when it happens to step on a door threshold. An agent stepping on a door threshold will exit the room if its current ideal distance to the door is 0, even if its tendency is *NOEXIT;* it will not exit the room if its tendency is *NOEXIT* and its ideal distance to the door is not zero at that moment. If there are no doors, then this parameter is dummy. Default is *NOEXIT*.

Parameters in the *AGENT* command can be specified progressively. Initial coordinates are mandatory, but the other parameters may be omitted. For example, we could specify initial coordinates, initial heading, and attention scope angle only:

AGENT 30 40 0 180

In this case, missing parameters will be assigned their default values.

If the *NAGENT* command is used, then the number of times that *AGENT* command is specified subsequently cannot be greater than the number of agents. If the *AGENT* command is specified a number of times less than the number of agents, then unspecified agents will be assigned their default parameter values. When graphical output is requested (see *OUTPUT*), icons of 5 by 5 pixels are used for representing agents on the screen. Different types of icons are available (see *ICON*).

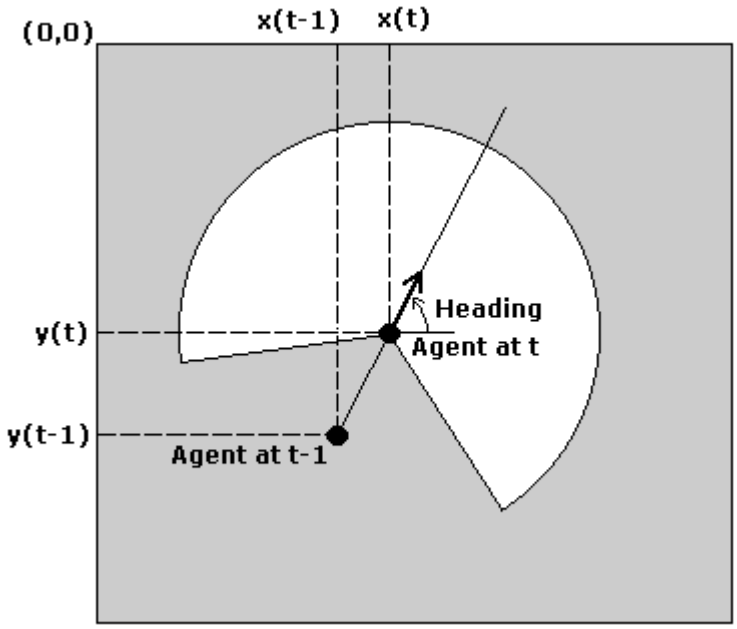


Figure 2. Coordinates, heading, and attention scope of an agent moving around the room.

BLOCK ***n*** **{** ***heading angle limit timeout***
IN | OUT SAME | RANDOM EXIT | NOEXIT }

Defines identical features for a block of *n* agents. Initial coordinates will of course be different for the agents in the block. If they are initially *IN* the room, then their coordinates will be assigned randomly. Parameter default values are the same as the ones for the *AGENT* command. Total number of agents, specified in one or more blocks, plus total number of times that *AGENT* is specified cannot exceed the number of agents defined with *NAGENT* or the maximum number of agents that is permitted.

NOBJECTS ***n***

Specifies the number of objects in the room. They are all rectangular. When graphical output is requested (see *OUTPUT*), each object is represented by a different color. Maximum number of objects is 5 and minimum is 0. Default is 0. To define objects, simply use as many *OBJECT* commands as necessary, and omit *NOBJECTS*. Nevertheless, if *NOBJECTS* is used, then it must precede any *OBJECT* command.

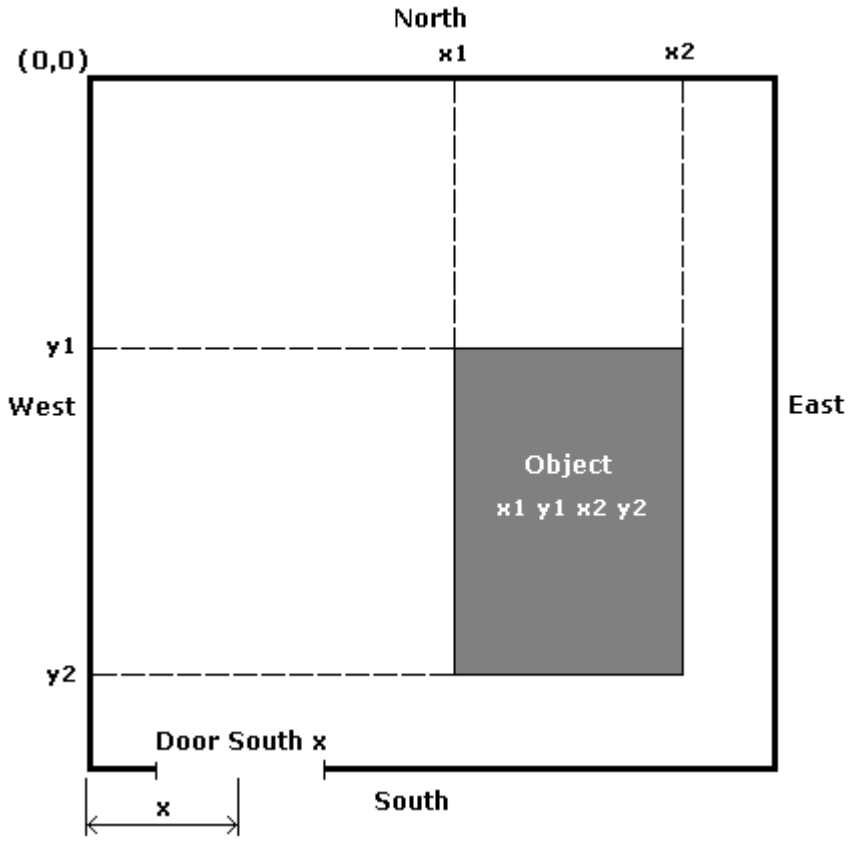


Figure 3. Object coordinates and door position.

OBJECT *x1 y1 x2 y2*

An object is defined by specifying coordinates for its northwest (*x1,y1*) and southeast (*x2,y2*) corners (see Figure 3). Objects are areas forbidden to agents, but agents can detect each other even if interposed objects exist between them. Overlapped objects are not permitted. Objects with corners outside of the room are trimmed off by the program. Real distance between an agent and a object is defined as the minimum of the distances from the agent to the four corners of the object, provided that corners fall within the agent´s attention scope. Real distance between two agents when objects are interposed between them is calculated as the shortest of all possible paths surrounding the objects and connecting the two agents. If *NOBJECTS* was specified previously, then the *OBJECT* command must be used as many times as necessary.

NDOORS *n*

Specifies the number of doors in the room. When graphical output is (see *OUTPUT*), doors are represented on the room walls using red lines. Maximum number of doors is 6 and minimum is 0. Default is 0. To define doors, simply use as many *DOOR* commands as necessary, and omit *NDOORS*. Nevertheless, if *NDOORS* is used, then it must precede any *DOOR* command.

DOOR { NORTH | SOUTH | EAST | WEST } *position*

A door is defined by specifying a wall (*NORTH, SOUTH, EAST, WEST*) and a position or coordinate for its center. All doors are 10 cells wide (see Figure 3). For example*, DOOR NORTH 20* is a door in the north wall, centered at 20 cells from the west wall. *DOOR WEST 20* is a door in the west wall, centered at 20 cells from the north wall. Real distance between an agent and a door is defined as the distance from the agent to the center of the door, provided that it falls within the agent´s attention scope. If *NDOORS* was specified previously, then the *DOOR* command must be used as many times as necessary.

GRAVITY *g* [*agent* | ALL]

Defines gravity *g* for an agent or for all of them. Agents are identified by their numbers (1, 2, ...). A single gravity may be defined for all agents by specifying *ALL*. As many *GRAVITY* commands may be used as agents have been previously defined. Agent numbers used in this command must refer to existing agents. Minimum gravity is 0.0 and maximum is 1.0. Default is 1.0. The greater the gravity, the higher the weighting of the differences between real and ideal distances on dissatisfactions. The smaller the gravity, the lower the weighting. When gravity is high, the effect of those differences on dissatisfaction depends very little on the real distances. When gravity is low, real distances must be small for the differences between real and ideal distances to affect dissatisfaction.

CHAOS *p*

Specifies the probability *p* that the agents choose random positions within their neighborhoods at each time unit, instead of those positions that minimize their dissatisfactions. If *p* = 0, then positions are never chosen randomly. Default is 0. When the program is running, chaos can be inserted simultaneously in all the agents by typing the hot key *C.* A magenta vertical line is displayed on the rectangular plot (see *RECTANGLE*) indicating the time unit when the key was pressed.

INERTIA *p*

Specifies the degree of inertia or constancy *p* ($0 \le p \le 1$) in the direction of movement when an agent is neutral towards evey other agent (see *IDEAL*). If inertia equals 1, then the agent will tend to move following its initial heading (if it hits a wall, its heading changes). If inertia equals 0, then the agent chooses a random heading at each time unit. Inertia works when agents are non-neutral as well, although in that case it may be masked by movements caused by changes in ideal distances. Inertia affects the agent decision as to which cell in the neighborhood is chosen when several cells have identical and minimum ideal dissatisfaction: If inertia is 1, then that cell having the minimum difference in heading relative to the current position will be chosen. If inertia is less than 1, then a probability (equal to $1-p$) exists that a cell is randomly chosen among those sharing the minimum dissatisfaction.

NEIGHBORHOOD [CIRCULAR | MOORE | NEUMANN | MVONN] [3 | 5 | 7 | 9 | 11 | 13 | 15]

Defines the agents' neighborhood, a region of cells surrounding each agent at time *t* and limiting the area where it can move at time *t + 1*. At time *t + 1,* the new neighborhood surrounds the agent's new position. Neighborhoods are usually symmetrical around the agent's position. Size and shape of neighborhood are the same for all agents. Neighborhoods of different agents can overlap. Default neighborhood is *CIRCULAR 3.* Long neighborhood diameters cause faster agent movements, and prevent that agents block each other at room corners. In MOORE and MVONN neighborhoods, movements can be in vertical, horizontal and diagonal movements; in a NEUMANN one, they can only be in vertical and horizontal directions; in a CIRCULAR one, movements are isotropic (see Figure 4).

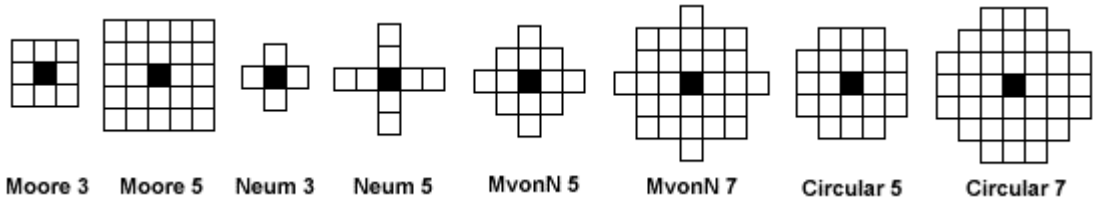


Figure 4. Some examples of neighborhoods. Black cells indicate the agent's current position.

GEOMETRY [EUCLIDEAN | CITYBLOCK | MAXIMUM]

Specifies the geometry for the two-dimensional space where agents move. In an *EUCLIDEAN* geometry, distance between two points *i* y *j* with coordinates (x_i, y_i), (x_j, y_j) (for example, two agents) is defined as:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

In a *CITYBLOCK* geometry, it is defined as:

$$d_{ij} = \left|x_i - x_j\right| + \left|y_i - y_j\right|$$

And in a *MAXIMUM* geometry, it is defined as:

$$d_{ij} = \max\left(\left|x_i - x_j\right|, \left|y_i - y_j\right|\right)$$

If agents were points with no dimension and the space they move around were really continuous, then the appropriate geometry would be *EUCLIDEAN*. In that case, an agent neighborhood would be a circle. However, as agents are located on and move around discrete cells, their actual neighborhoods and the space geometry are approximations. When neighborhood is *CIRCULAR* and its diameter is large, the euclidean distance between two agents approaches the shortest path between them. When neighborhood is *NEUMANN,* the city-block distance between two agents equals the shortest path between them. When neighborhood is *MOORE* or *MVONN,* the maximum distance between two agents approches the shortest path between them. Default geometry is *EUCLIDEAN.* The maximum possible distance in the room (*maxdist*) equals the length of the room's diagonal when geometry is *EUCLIDEAN*, the sum of its sides when geometry is *CITYBLOCK*, and its longer side when geometry is *MAXIMUM*.

ICON [FACE | CROSS | CIRCLE | HEADING | READINESS]

Defines the icon used to represent agents when graphical output is requested (see *OUTPUT*). Default value is *FACE*. When *FACE, CROSS,* or *CIRCLE* are specified, a single type of icon is used for representing the agents, both when they are inside and outside of the room, and also for identifying them in other graphs (see *SQUARE* and *RECTANGLE*). When *ORIENTATION* is specified, agents in the room are represented by arrows showing their approximate headings (with a precision of 45 degrees), while agents outside of the room are represented by arrows pointing east. When *TENDENCY* is specified, solid and empty circles are used for representing agents with *EXIT* and *NOEXIT* readinesses, respectively. Icons shown in the *SQUARE* and *RECTANGLE* plots are surrounded by arcs showing the areas falling out of the agent's attention scope angle. When an agent is out of the room, its icon is shown left to the west wall, and the

time left until the agent will reenter the room is represented by an arc surrounding the icon. This command is dummy when no graphical output is requested (see *OUTPUT*).

Icon colors are repeated for agents 1 and 11 (green), 2 and 12 (red), and so forth. Two slightly different icon subtypes are used for distinguishin agents 1 thru 10 from agents 11 thru 20:

Icon type	*Agents 1 thru 10*	*Agents 11 thru 20*
ICON FACE		
ICON CROSS		
ICON CIRCLE		
ICON HEADING		
ICON READINESS		
If EXIT		
If NOEXIT		

SQUARE [DISTANCE { *agent1 agent2 …* } |
PHASE { DISTANCE | DISSATISFACTION | FRUSTRATION | ENCODED
lag **LINK | NOLINK }]**

Specifies the graph to be shown on the lower right square area of the screen. *DISTANCE* indicates that a matrix of ideal and real distances must be plotted against time for every agent-to-agent interaction. By default, real and ideal interagent distances are plotted against time for agents 1 thru *n*, with *n* £ 10. Specific agents can be selected by number.

PHASE indicates that a certain variable at time *t* must be plotted against its value at time *t* - *lag.* Phase plots for the mean real distance among agents, their mean real dissatisfaction, their mean frustration, or their encoded frustration can be requested. An agent's frustration at time *t* is defined as the difference between its real and ideal dissatisfactions at *t*. Encoded frustration is an integer between 0 and 2^n-1, where *n* is the number of agents. At time *t*, the encoded frustration is computed as:

$$f(n-1)\cdot 2^{n-1} + f(n-2)\cdot 2^{n-2} + \ldots + f(1)\cdot 2^{1} + f(0)\cdot 2^{0}$$

where $f(k-1)=1$ if agent *k*'s frustration is positive at *t*,

$f(k-1)=0$ if agent *k*'s frustration is negative or zero at *t*.

Lag is an integer between 1 and 20; default is 1. When *LINK* is requested, consecutive data points are linked by a straight line. Default is *NOLINK.* This command is dummy if no graphical output is requested (see *OUTPUT*), or if room width is greater than 80 (as it prevents the square graph to be displayed).

RECTANGLE [[DISSATISFACTION | DISTANCE | FRUSTRATION] { *agent1 agent2 …* }
| AGGREGATION { *maxneighbor* }]

Specifies the graph to be shown on the lower left rectangular area of the screen. Agent's real dissatisfaction, its mean real distance from other agents, its frustration, or aggregation indexes can be plotted against time. By default, dissatisfactions are plotted. When *DISSATISFACTION, DISTANCE,* or *FRUSTRATION* are requested, specific agents can be selected by number. If agent numbers are omitted, then dissatisfactions, distances or frustrations are plotted for every agent. If the first agent number is zero, then the mean value (±2 standard deviations) of the requested variable for all agents is plotted against time. At time *t*, only variables for agents in the room at that moment are plotted.

If *AGGREGATION* is requested, then Clark-Evans' nearest neighbor index is plotted against time, with its 95% non-significant bounds. This index is distributed as a normal standard score. When the agents are distributed randomly in space, the index is non significant. When the agents are clustered, it is negative and significant. Finally, when the agents are distributed regularly or uniformly in space, it is positive and significant. First to fifth nearest neighbors aggregation indexes can be requested. For example, *RECTANGLE AGGREGATION 3* plots the first, second, and third nearest neighbors aggregation indexes. This command is dummy if no graphical output is requested (see *OUTPUT*), or if room height is greater than 70 (as it prevents the rectangular graph to be displayed).

DISSATISFACTION *minimum maximum*

Defines minimum and maximum values to display when dissatisfactions are plotted on screen. Default values are 0.0 and 1.0. This command is dummy if no graphical output is requested (see *OUTPUT*).

DISTANCE *minimum maximum*

Defines minimum and maximum values to display when ideal or real distances are plotted on screen. Default values are 0.0 and *maxdist* (see *GEOMETRY*). This command is dummy if no graphical output is requested (see *OUTPUT*).

FRUSTRATION *minimum maximum*

Defines minimum and maximum values to display when frustrations are plotted on screen. Default values are –1.0 and +1.0. This command is dummy if no graphical output is requested (see *OUTPUT*).

AGGREGATION *minimum maximum*

Defines minimum and maximum values to display when aggregation indexes are plotted on screen. Default values are –3.0 and +3.0. This command is dummy if no graphical output is requested (see *OUTPUT*).

PAUSE *p*

Specifies the pause factor $p > 0$ used by the program to slow down the process when only graphical output is requested. High values of p make the process slow. In old computers, p must be set to low values. Defining many agents and objects tends to make the program slower as well. This command is dummy if no graphical output is requested (see *OUTPUT*).

TRAJECTORY [YES | NO]

Specifies whether agents' trajectories around the room must be displayed. Default is NO. This command is dummy if no graphical output is requested (see *OUTPUT*).

CELLS [YES | NO]

Specifies whether the grid of cells must be displayed on the room's background. Default is NO. This command is dummy if no graphical output is requested (see *OUTPUT*).

OUTPUT [GRAPHICAL | FILE | ALL] { PERIOD | COMMA }

Specifies the output mode. If *GRAPHICAL* is requested, then the program shows the position of the agents in the room as time progresses. If room width is less than or equal to 80, then both a map showing frequency of occupation of the room cells, and a plot requested with the *SQUARE* command are represented on the right side of the screen. If room height is less than or equal to 70, then a plot requested with the *RECTANGLE* command is shown below the room. If *OUTPUT GRAPHICAL* is specified, then the rest of the parameters in this command are dummy. *OUTPUT GRAPHICAL* is the default specification.

When *OUTPUT FILE* is specified, no graphical output is displayed and numerical results are saved in a file. Output file has the same name as the command file, but extension is TXT. Rows and columns in a TXT file correspond to time units and variables, respectively. Values are separated with tabs, and thus the file can be read by SPSS, StatGraphics, or any spreadsheet. Results saved are those specified with the *VARIABLES* command. *PERIOD* or *COMMA* specify the punctuation sign for decimals to be used in the output file. Default punctuation is *PERIOD.*

When *OUTPUT ALL* is specified, the program will display graphical output and simultaneously save results in a file. Format and decimal punctuation for the output file can be specified as well.

The program always saves a file with extension LOG containing an interpretation of the commands and a description of the results that are saved in the TXT file.

VARIABLES [ALL | COOR:IDIS:RDIS:VIDI:VLIM:AGGR:PERM]

Specifies the variables to be saved in the output file. By default, all variables are saved. This command is dummy if no output file is requested. Variables are:

COOR Agent coordinates
IDIS Agent ideal dissatisfaction (expected before moving)
RDIS Agent real dissatisfaction (obtained after moving)
VIDI Variable ideal distances between agents (if models with variable ideal distances have been specified)
VLIM Personal and social space for agents with Evolutionary models, and time limits for stagnation and exile periods for agents with Hybrid models (see IDEAL)
AGGR First to fifth neighbor aggregation indexes (or less, if there are less than six agents defined)
PERM Agent permutation order

IDEAL [USER | NEUTRAL | GROUP { *d* } | SUBGROUPS { *d1 d2* } |
SEPARATE { *d* } | CULTURAL { *d1 d2* } | RANDOM { *d1 d2* } |
CHASE { *d1 d2* } | CCHASE { *d1 d2* } | LCHASE { *d1 d2* } |
UNSOCIABLE { *d1 d2 t* } | SOCIABLE { *d1 d2 t* } | FATIGUE { *d1 lt ld d2 t* } |
DEPRIVATION { *d1 lt ld d2 t* } | COMPLEX { *d1 lt ld d2 t* } |
EVOLUTIONARY { *lt cf* } | HYBRID { *cd lt cf* }]

This command is mandatory, and the only one that is necessary. *IDEAL* defines a matrix of agent-to-agent, and also agent-to-object and agent-to-door, ideal distances. If no objects or doors have been defined, it is an n x n matrix, where n is the number of agents. Otherwise, its dimensions are n x ($n+b+d$), where b and d are the number of objects and doors, respectively. Matrix cells contain constant or variable ideal distance models, which are specified with a word and a series of numerical parameters. Cell (i,j) contains the ideal distance that agent i wishes to keep from agent j (if $j \leq n$), from object j (if $n < j \leq n+b$), of from door j (if $n+b < j \leq n+b+d$). If no agents have been defined, number of rows in the matrix will equal 10, which is the default number of agents (see *NAGENTS*), and number of columns will equal $10+b+d$.

IDEAL USER

Indicates that ideal distance models for all cells in the matrix will be specified by the user subsequently. The matrix must be completely specified in the lines immediately following this command. Ideal distances are always greater than or equal to zero. When ideal distance between agents equals zero, they be can eventually in contiguous cells but never in the same cell. Likewise, when an agent-to-object ideal distance is zero, the agent can be eventually next to the object, but never on the object. However, when an agent-to-door ideal distance is zero, the real distance can be eventually zero, in which case the agent will exit the room. The following models are possible:

Model	*Parameters*	*Example*
Quiet	-	Q
Neutral	-	N
Constant	*value*	C 3
Random	*lower upper*	R 1 10
Unsociable	*start end range*	U 1 30 100
Sociable	*start end range*	S 50 3 100
Fatigue	*start time_limit distance_limit end range*	F 2 20 2 50 100
Deprivation	*start time_limit distance_limit end range*	D 50 30 50 1 100
Evolutionary	*time_limit change_factor*	E 10 0.01
Hybrid	*critical_distance time_limit change_factor*	H 6 10 0.01

Only the first letter is required for specifying the model name.

Model *Q*

It can be assigned to diagonal cells *(i,i)* in the matrix to indicate that agent *i* will not move during the simulation. This specification will cancel any agent parameter which may affect of be related to its movement (for example, his tendency to exit the room; see *AGENT*).

Model *N*

It indicates that agent *i* is indifferent to agent, object or door *j*, that is, that the real distance between them will not affect agent *i*'s dissatisfaction. If *Q* is specified for a non-diagonal cell, then the program will automatically assign *Q* to the diagonal cell in that same row, and *N* to all non-diagonal cells in the row.

Model *C value*

It specifies that agent *i* wants to keep a constant distance equal to *value* from agent, object, or door *j*.

Model *R lower upper*

It specifies that agent *i* wants to keep a distance from agent, object, or door *j* that varies randomly and uniformly between *lower* and *upper* at each time unit.

Model *U start end range*

It specifies that agent *i* wants to keep a distance from agent, object, or door *j* that increases linearly from *start* to *end* during the first time units of the simulation, as specified by *range.* When the ideal distance reaches *end*, it will remain constant at that value for the rest of the simulation.

Model *S start end range*

It specifies that agent *i* wants to keep a distance from agent, object, or door *j* that decreases linearly from *start* to *end* during the first time units of the simulation, as specified by *range.* When the ideal distance reaches *end*, it will remain constant at that value for the rest of the simulation.

Model *F start time_limit distance_limit end range*

It specifies that agent *i* wants to keep a distance from agent, object, or door *j* that will be constant and equal to *start*. However, if real distance from agent, object, or door *j* is lower than *distance_limit* for more than *time_limit* consecutive time units, then ideal distance will increase linearly from *start* to *end* values during the time units specified by *range*. When *range* is reached, the ideal distance regains its *start* value abruptly. The ideal distance will change again when critical conditions described by *distance_limit* and *time_limit* occur.

Model D start time_limit distance_limit end range

It specifies that agent *i* wants to keep a distance from agent, object, or door *j* that will be constant and equal to *start*. However, if real distance from agent, object, or door *j* is greater than *distance_limit* for more than *time_limit* consecutive time units, then ideal distance will decrease linearly from *start* to *end* values during the time units specified by *range*. When *range* is reached, the ideal distance regains its *start* value abruptly. The ideal distance will change again when critical conditions described by *distance_limit* and *time_limit* occur.

Model *E time_limit change_factor*

It indicates that the ideal distance from agent *i* to agent *j* is not set initially but evolves or is generated as the simulation progresses. Initially, lower and upper limits for distances are given ("personal" and "social" spaces, respectively). The former equals the neighborhood diameter, and the latter is three times bigger. If real distance from *i* to *j* is less than personal space, then a time counter is updated. When the counter exceeds *time_limit*, personal space from *i* to *j* decreases and the counter is reset; otherwise, personal space increases. In any case, ideal distance is set equal to the updated personal space. Initially, if real distance is greater than

social space, agent *i* acts as if it were neutral towards agent *j,* until real distance becomes less than social space.

If real distance is greater than social space, then a time counter is updated. When the counter exceeds *time_limit*, social space from *i* to *j* increases and the counter is reset; otherwise, social space decreases. In any case, ideal distance is set equal to the updated social space.

If real distance is between personal and social spaces, it is set equal to real distance, and a time counter is updated. When the counter exceeds *time_limit*, personal and social spaces decrease and the counter is reset.

Whenever personal and social spaces change, their increase or decrease is equal to *change_factor* multiplied by the neighborhood diameter. If personal space increases so that it exceeds social space, then the latter is increased as well. If social space decreases so that it is less than personal space, then social space either remains unchanged, or set equal to the current personal space plus *change_factor* multiplied by the heighborhood diameter, whichever is smaller.

Model *H critical_distance time_limit change_factor*

Like the Evolutionary model, ideal distance is computed from the interactions between agents. When a hybrid model is specified for the agent *i* to agent *j* interaction, agent *i* initially acts as if it were neutral to agent *j*, while their real distance is greater than *critical_distance*.

When real distance is less than *critical_distance*, the model is 'activated' for agent *i*. Since that moment, its ideal distance to agent *j* undergoes two different kinds of change. A smooth change is caused by agent *i* adapting to agent j's movements: Ideal distance decreases whenever agent *i* tries to approach agent *j*, and receives a negative partial fustration from it (that is, agent *i* gets, or approaches, more than predicted), or whenever agent *i* tries to leave agent *j*, and receives a positive partial frustration from it (that is, it gets, or leaves, less than predicted). Ideal distance increases whenever agent *i* tries to leave and partial frustration is negative, or when it tries to approach and partial frustration is postive. If partial frustration is null, ideal distance does not change. If agent *i* does not try to approach or leave agent *j*, then ideal distance increases or decreases with probability 1/2. Smooth change in ideal distance is equal to *change_factor.*

Abrupt changes in ideal distance occur when stagnation periods of smooth changes are reached, for example when ideal distance remains constant, or when cyclic changes appear (increase / decrease / increase / decrease /...), for *time_limit* time units. In those cases, ideal distance is increased by *time_limit***change_factor*, and agent *i* is 'exiled' for *time_limit* time units. When exiled, ideal distance remains unchanged. When exile time is over, ideal distance is

decreased by *time_limit*change_factor* + *critical_distance*, and the smooth change phase is resumed. The time limit defining stagnation is increased (by *change_factor*) when an abrupt change occurs, and the time limit defining exile is decreased (by *change_factor*) when the smooth phase is resumed.

This model is called hybrid because it includes two different mechanisms that cause changes in the ideal distance, and also because agents behaving according to it are 'fatigued' (by stagnation) and 'deprived' (by exile).

Example (for 3 agents and 1 object)

```
Ideal user
N            U 1 30 100   U 1 40 500   N
S 50 3 100   N            C 5          C 30
C 1          C 50         N            N
```

Agent 1 wants to increase its distance from agents 2 and 3 (ideal distances increasing from 1 to 30 in 100 time units, and from 1 to 40 in 500 time units, respectively). Agent 2 wants to decrease its distance from agent 1 (from 50 to 3 in 100 time units), while keeping a constant distance 5 from agent 3. Agent 3 wants to keep constant distances 1 and 50 from agents 1 and 2, respectively. Agents 1 and 3 are neutral towards the object, but agent 2 wants to keep a constant distance 30 from it.

General models for the group of agents as a whole can be specified. They are automatically built up by the program by assigning certain ideal distance models to each agent-to-agent accordingly. In that case, the program assigns *N* to agent-to-object and agent-to-door cells if objects or doors are defined. General models are described below:

IDEAL NEUTRAL

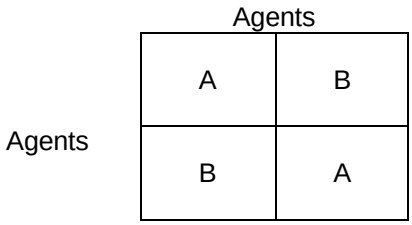
Indicates that *N* must be assigned to all agent-to-agent cells. As a consquence, real distances will not affect dissatisfactions.

IDEAL GROUP { d }

Indicates that *C d* must be assigned to all non-diagonal agent-to-agent cells, where $d \geq 0$, and *N* to all diagonal cells. Default is $d = 2$. A single group of agents will be formed. If $d < 0$, then the program assigns *C d* to non-diagonal cells, where *d* is a random value independently sampled for each cell from a normal distribution with mean *abs(d)* and standard deviation *abs(d)/4.*

IDEAL SUBGROUPS { d1 d2 }

Indicates that *C d1* and *C d2* must be assigned to all non-diagonal agent-to-agent cells in submatrices A and B, respectively:



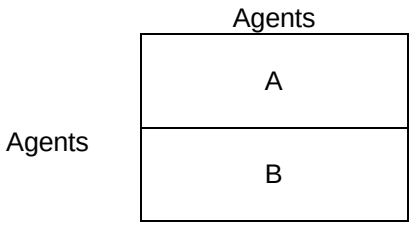
If the number of agents is even, then the number of rows in A and B are identical. Otherwise, A has one more row than B. Defaults are *d1 = 2, d2 = maxdist/4* (see *GEOMETRY*). Two different groups of agents will be formed. If *d1* < 0, then the program assigns *C d1* to non-diagonal cells in submatrices A, where *d1* is a random value independently sampled for each cell from a normal distribution with mean *abs(d1)* and standard deviation *abs(d1)/4.* If *d2* < 0, then the program assigns *C d2* to non-diagonal cells in submatrices B, where *d2* is a random value independently sampled for each cell from a normal distribution with mean *abs(d2)* and standard deviation *abs(d2)/4.*

IDEAL SEPARATE { d }

It is equivalent to *IDEAL GROUP { d }*, but the default is *d = (3/4)*maxdist* (see *GEOMETRY*) When this default is used, agents tend to scatter.

IDEAL CULTURAL { d1 d2 }

Indicates that *C d1* and *C d2* must be assigned to all non-diagonal agent-to-agent cells in submatrices A and B, respectively:



If the number of agents is even, then the number of rows in A and B are identical. Otherwise, A has one more row than B. Defaults are *d1 = 2, d2 = maxdist/4* (see *GEOMETRY*). Some agents will tend to form a group, while some others will tend to scatter. If negative values are specified for *d1* or *d2,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL RANDOM { d1 d2 }

Indicates that *R d1 d2* must be assigned to all non-diagonal agent-to-agent cells. Defaults are *d1 = 2* and *d2 = maxdist/4* (see *GEOMETRY*). If *d1* < 0, then the program assigns *d1* as the lower limit of the uniform distribution, where *d1* is a random value independently sampled for each cell from a normal distribution with mean *abs(d1)* and standard deviation *abs(d1)/4.* If *d2* < 0, then the program assigns *d2* as the upper limit of the uniform distribution, where *d2* is a random value independently sampled for each cell from a normal distribution with mean *abs(d2)* and standard deviation *abs(d2)/4.* For example, *IDEAL RANDOM -5 -20* will assign *R a b* to each non-diagonal agent-to-agent cell independently, with *a* → **N**(5,5/4*), b* → **N**(20,20/4).

IDEAL CHASE { d1 d2 }

Indicates that *C d1* and *C d2* must be assigned to all non-diagonal A and B agent-to-agent cells, respectively:

Agents						
Agents	-	A	B	B	…	B
	B	-	A	B	…	B
	B	B	-	A	…	B
	B	B	B	-	…	B
	…	…	…	…	…	…
	A	B	B	B	B	-

Defaults are *d1 = 2, d2 = maxdist/4* (see *GEOMETRY*). Agents will chase each other forming a stressed circle, with agent *i* approaching agent *i*+1 (and agent *n* approaching agent 1) and moving away from all other agents. If negative values are specified for *d1* or *d2,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL CCHASE { d1 d2 }

Indicates that *C d1* and *C d2* must be assigned to all non-diagonal A and B agent-to-agent cells, respectively:

Agents						
Agents	-	A	*N*	*N*	…	B
	B	-	A	*N*	…	*N*
	N	B	-	A	…	*N*
	N	*N*	B	-	…	*N*
	…	…	…	…	…	…
	A	*N*	*N*	*N*	B	-

Defaults are *d1 = 2, d2 = maxdist/4* (see *GEOMETRY*). Agents will chase each other forming a flexible circle, with agent *i* approaching agent *i*+1, and agent *i*+1 moving away from agent *i* (and agent *n* approaching agent 1, and agent 1 moving away from agent *n*). If negative values are

specified for *d1* or *d2,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL LCHASE { d1 d2 }

Indicates that *C d1* and *C d2* must be assigned to all non-diagonal A and B agent-to-agent cells, respectively:

Agents						
Agents	-	A	*N*	*N*	…	B
	B	-	A	*N*	…	*N*
	N	B	-	A	…	*N*
	N	*N*	B	-	…	*N*
	…	…	…	…	…	…
	N	*N*	*N*	*N*	B	-

Defaults are *d1 = 2, d2 = maxdist/4* (see *GEOMETRY*). Agents will chase each other forming a line, with agent *i* approaching agent *i*+1, and agent *i*+1 moving away from agent *i* (and agent 1 moving away from agent *n*, but agent *n* not approaching agent 1). If negative values are specified for *d1* or *d2,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL UNSOCIABLE { d1 d2 t }

Indicates that *U d1 d2 t* must be assigned to all non-diagonal agent-to-agent cells. Defaults are *d1 = 2, d2 = maxdist/4, t = time/2*, where *time* is the total number of steps or discrete time units for the simulation (see *GEOMETRY*). Agents become progressively more unsociable, i.e., they tend to move away from each other as the simulation progresses. If negative values are specified for *d1, d2*, or *t,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL SOCIABLE { d1 d2 t }

Indicates that *S d1 d2 t* must be assigned to all non-diagonal agent-to-agent cells. Defaults are *d1 = 2, d2 = maxdist/4, t = time/2*, where *time* is the total number of steps or discrete time units for the simulation (see *GEOMETRY*). Agents become progressively more sociable, i.e., they tend to approach each other as the simulation progresses. If negative values are specified for *d1, d2*, or *t,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL FATIGUE { d1 lt ld d2 t }

Indicates that *F d1 lt ld d2 t* must be assigned to all non-diagonal agent-to-agent cells. Defaults are *d1 = 2, lt = time/20, ld = maxdist/4, d2 = maxdist/3, t = time/10*, where *time* is the

total number of steps or discrete time units for the simulation (see *GEOMETRY*). Agents are fatigated towards each other when critical conditions for social fatigue occur. If negative values are specified for *d1, lt, ld, d2,* or *t,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL DEPRIVATION { d1 lt ld d2 t }

Indicates that *D d1 lt ld d2 t* must be assigned to all non-diagonal agent-to-agent cells. Defaults are *d1 = maxdist/3, lt = time/20, ld = maxdist/4, d2 = 2, t = time/10*, where *time* is the total number of steps or discrete time units for the simulation (see *GEOMETRY*). Agents are deprivated towards each other when critical conditions for social deprivation occur. If negative values are specified for *d1, lt, ld, d2,* or *t,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL COMPLEX { d1 lt ld d2 t }

Indicates that *D d1 lt ld d2 t* and *F d2 lt ld d1 t* must be assigned to all non-diagonal A and B agent-to-agent cells, respectively:

Agents						
Agents	-	A	B	B	…	B
	B	-	A	B	…	B
	B	B	-	A	…	B
	B	B	B	-	…	B
	…	…	…	…	…	…
	A	B	B	B	B	-

Defaults are *d1 = maxdist/3, lt = time/20, ld = maxdist/4, d2 = 2, t = time/10*, where *time* is the total number of steps or discrete time units for the simulation (see *GEOMETRY*). Agent *i* is socially deprived from agent *i*+1 (and agent *n* from agent 1), but socially fatigated towards all other agents. The group as a whole will display a global behavior of approaching and moving away with different cyclicities for different agent-to-agent interactions. If negative values are specified for *d1, lt, ld, d2,* or *t,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL EVOLUTIONARY { lt cf }

All agents generate their ideal distance to every other agent in an evolutionary way, using time limit *lt* and change factor *cf* in order to adapt their personal and social spaces to every other agent and, as a consequence, their ideal distances. By default*, lt = time/20, cf = 0.01*. If negative values are specified for *lt* and *cf,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

IDEAL HYBRID { cd lt cf }

All agents generate their ideal distance to every other agent according to the hybrid model, using critical distance *cd*, initial time limit for stagnation and exile *lt*, and change factor *cf*. By default, *dc = neighborhood_diameter, lt = time/20, fc = 1.0.* If negative values are specified for *cd, lt* and *cf,* then the program assigns random values independently sampled for each cell from a normal distribution accordingly (see *IDEAL SUBGROUPS*).

Command order

Commands can be specified in any order, except in the following cases:

- In order to define a number of agents that is different from the default one (10), or to define objects and doors, *NAGENTS, AGENT, NOBJECTS, OBJECT, NDOORS,* and *DOORS* must be specified before *IDEAL.*
- In order to use numbers identifying agents in *GRAVITY, SQUARE* or *RECTANGLE*, agents must be defined previously.
- If some commands are repeated, then parameters are overriden. For example, if *GRAVITY 0.5* is specified and, later on in the file, *GRAVITY 0.8* is, 0.8 overrides 0.5.

We recommend that commands are specified in the order they are described in this manual.

Examples

- The following commands define six agents with default parameters, and one object. The ideal distance matrix is generated by the program. Only graphical output is requested. By default, real dissatisfactions are plotted against time, and a lag 10 phase plot of the mean dissatisfaction is shown. In both plots, only dissatisfactions between 0 and 0.20 are displayed:

```
nsimul 20
time 500
room 80 70
nagents 6
object 30 30 40 50
gravity 0.02 all
dissat 0 0.20
square phase dissat 10 link
ideal chase
```

- The following commands define two agents initially in the room, one object, and two doors. Agents are neutral to each other, but both want to exit the room through a door different from the one to which they are close initially. Once they exit the room, they re-enter it through a random door:

```
nsim 5
time 10000
room 60 60
grav 0.5 all
agen 3 10  -1 360 50 10 in
agen 50 52 -1 360 50 20 in
obje 0 15 45 60
door west 7
door south 50
ideal user
n  n  n  n   c 0
n  n  n  c 0 n
```

- The following commands define four agents with different gravities and attention scopes. Some ideal distances are constant, while some vary according to time in proximity:

```
time 2000
room 120 90
block 2 –1 360
block 2 –1 90
gravity 0.20 1 2
gravity 0.80 3 4
```

```
ideal user
n              c 4            d 10 10 10 1 50  n
c 30           n              d 10 10 10 1 50  n
f 1 5 1 30 10  f 1 5 1 30 10  n                c 30
f 1 5 1 30 10  f 1 5 1 30 10  c 4              n
```

- The following commands define fifteen agents with maximum attention scope. All of them are initially in the room. Two objects and three doors are defined. Both graphical and numerical output are selected. As no *VARIABLES* have been specified, all variables will be saved in the output file. Ideal distances are generated by the program according to the "complex" global model:

```
nsim 5
time 1000
room 60 60
block 15 -1 360 1000 10 in same exit
gravity 0.2 all
obje 10 15 30 35
obje 30 30 50 50
door north 20
door west  20
door south 20
neighborhood circular 5
rectangle dissat 0
square phase dissat 1
output all
cells yes
ideal complex
```

Output

P-Space always saves command interpretation and a description of the TXT file, in a file with extension LOG:

```
P-SPACE 3.3.2  February 2000  Spatial behavior simulator
Computer Technology in the Behavioral Science Research Group

Thu Feb 24 17:04:06 2000

Command file: DATA\ROBO22

> nsim 5
> time 10000
> room 60 60
> grav 0.5 all
> squa phase diss 10 link
> obje 0 15 45 60
> agen 3 10  -1 360 80 10 in
> agen 50 52 -1 360 80 20 in
> door west 7
> door south 50
> outp all
> ideal user
> n n n  n   c 0
> n n n  c 0 n

Command interpretation:

Simulations         5
Time per simulation 10000
Room                60 x 60
Neighborhood        Circular 3
Geometry            Euclidean
Chaos               0.0000
Inertia             1.0000
Agents              2
Objects             1
Doors               2
Gravity             0.5000

 Agent Coordinates  Orient    Scope  Maximum  T_Out IniPlace Reentry   Readiness
     1  (  3,  10)      -1      360       80   10.0 In       Random    Noexit
     2  ( 50,  52)      -1      360       80   20.0 In       Random    Noexit

Object Coordinates
     1 (  0,  15) ( 45,  60)

  Door Wall  Place
     1 West      7
     2 South    50

Ideal distances matrix:
Type: User
All agent-to-agent distances are neutral
All agent-to-object distances are neutral

Age Doo Model        Parameters
  1   2 Constant     Val=  0
  2   1 Constant     Val=  0

Coordinates:  -1, initial coordinates are random; -2, agent is out
Heading:      -1, initial heading is random

Output file:
Variables in output file:
 Simul     Simulation number
 Time      Time unit or simulation step
 XI YI     Agent I's coordinates
 IDiI      Agent I's ideal dissatisfaction
 RDiI      Agent I's real dissatisfaction
 PerI      Agent priority level I according to permutation
 AggK      K-th nearest neighbor standardized aggregation index
```

Rows and columns in the TXT file correspond to timue units and variables, respectively. The first row contains variable names are saved, and values are separated with tabs:

Simul	Time	X1	Y1	IDi1	RDi1	X2	Y2	IDi2	RDi2
1	1	4	11	1.00000	1.00000	49	51	0.98584	0.98584
1	2	3	10	1.00000	1.00000	48	50	0.97327	0.97327
1	3	3	11	1.00000	1.00000	47	49	0.96100	0.96100
1	4	4	11	1.00000	1.00000	47	48	0.94922	0.94922
1	5	3	10	1.00000	1.00000	47	47	0.93744	0.93744
1	6	3	10	1.00000	1.00000	47	46	0.92566	0.92566
1	7	2	10	1.00000	1.00000	47	45	0.91388	0.91388
1	8	1	11	1.00000	1.00000	47	44	0.90211	0.90211
1	9	0	10	1.00000	1.00000	47	43	0.89033	0.89033
1	10	0	11	1.00000	1.00000	47	42	0.87855	0.87855
1	11	0	11	1.00000	1.00000	47	41	0.86677	0.86677
1	12	0	10	1.00000	1.00000	47	40	0.85500	0.85500
1	13	0	11	1.00000	1.00000	47	39	0.84322	0.84322
1	14	1	10	1.00000	1.00000	47	38	0.83144	0.83144
1	15	2	11	1.00000	1.00000	47	37	0.81967	0.81967
1	16	1	11	1.00000	1.00000	47	36	0.80790	0.80790
1	17	0	11	1.00000	1.00000	47	35	0.79612	0.79612
1	18	1	11	1.00000	1.00000	47	34	0.78435	0.78435
1	19	1	10	1.00000	1.00000	47	33	0.77258	0.77258
1	20	0	10	1.00000	1.00000	47	32	0.76082	0.76082
1	21	0	11	1.00000	1.00000	47	31	0.74905	0.74905
1	22	0	12	1.00000	1.00000	47	30	0.73729	0.73729
1	23	1	12	1.00000	1.00000	47	29	0.72553	0.72553

Errors

The following error messages may be emitted by P-Space:

Error 1 Wrong number of simulations
Error 2 Wrong time
Error 3 IDEAL has been specified before NAGENTS
Error 4 Wrong number of agents
Error 5 IDEAL has been specified before AGENT
Error 6 Wrong agent specification
Error 7 Wrong agent initial heading
Error 8 Wrong agent attention scope
Error 9 Wrong agent maximu attention distance
Error 10 Wrong agent mean time out
Error 11 Wrong agent initial place
Error 12 Wrong agent re-entry door
Error 13 Wrong agent exit readiness
Error 14 Wrong agent initial coordinate
Error 15 Agent coordinates are not free or not possible
Error 16 IDEAL has been specified before BLOCK
Error 17 Wrong block specification
Error 18 Wrong block initial heading
Error 19 Wrong block attention scope
Error 20 Wrong block maximum attention distance
Error 21 Wrong block mean time out
Error 22 Wrong block initial place
Error 23 Wrong block re-entry door
Error 24 Wrong block exit readiness
Error 25 IDEAL has been specified before NOBJECTS
Error 26 Wrong number of objects
Error 27 IDEAL has been specified before OBJECTS
Error 28 Wrong object coordinates
Error 29 Objects intersect
Error 30 Wrong room dimensions
Error 31 Error when specifying ideal distances
Error 32 No ideal distances have been specified
Error 33 Wrong trajectory specification
Error 34 Wrong cell display specification
Error 35 Wrong output specification
Error 36 Wrong output variable specification
Error 37 No output variables have been selected

Error 38 Wrong pause specification
Error 39 Wrong gravity specification
Error 40 NAGENTS has not been specified before GRAVITY
Error 41 Undefined agent when specifying gravity
Error 42 Wrong neighborhood specification
Error 43 Wrong geometry specification
Error 44 Wrong chaos specification
Error 45 Wrong inertia specification
Error 46 Wrong icon specification
Error 47 Wrong dissatisfaction bounds specification
Error 48 Wrong distance bounds specification
Error 49 Wrong frustration bounds specification
Error 50 Wrong aggregation indexes bounds specification
Error 51 Wrong SQUARE specification
Error 52 Wrong SQUARE type
Error 53 Wrong agent number when specifying SQUARE
Error 54 Wrong phase plot type
Error 55 Wrong phase plot style
Error 56 Wrong RECTANGLE specification
Error 57 Wrong RECTANGLE type
Error 58 Wrong nearest neighbor number when specifying RECTANGLE
Error 59 Wrong agent number when specifying RECTANGLE
Error 60 IDEAL has been specified before NDOORS
Error 61 Wrong number of doors
Error 62 IDEAL has been specified before DOOR
Error 63 Wrong door wall
Error 64 Wrong command
Error 65 Could not open input file
Error 66 Could not open log file
Error 67 Could not open output file
Error 68 Could not open help file
Error 69 Could not run browser
Error 70 Could not initialize graphics mode